Document-Based Question Answering Using LLMs and RAG: A Prototype and Evaluation

Eddy Jiang

June 1, 2025

Literature Survey

Document-based question answering (DBQA) is an NLP task that involves developing systems to read unstructured documents and answer questions based on their content. It encompasses two subtasks: extractive QA (locating answers in the text) and abstractive QA (synthesizing new natural language answers).

Early solutions relied on supervised approaches fine-tuned on QA datasets like SQuAD (Chen et al., 2017; Seo et al., 2016), but performance was limited by the computational capacity of models at the time and the need for high-quality labeled data. The rise of transformer-based models like BERT improved performance, and the recent boom of LLMs, such as GPT and Llama, has enabled zero-shot DBQA with minimal additional training (Phogat et al., 2023).

However, LLMs still struggle with long documents that exceed their context windows, such as books and complex instructions (Fraga, 2024). To address this, retrieval-augmented generation (RAG) has emerged as a leading approach to DBQA (Lewis et al., 2020). RAG is a multi-step framework consisting of a retriever (typically a dense vector search algorithm) and a generator (usually an LLM). According to (Khan et al., 2024), it is performed as follows:

1. File input is broken into smaller chunks using a pre-defined strategy (e.g., by paragraph, section, or token window).
2. Chunks are converted into vector embeddings and stored in a database to support semantic search.
3. When receiving a query, the retriever fetches the most relevant chunks based on similarity to the user's question.

4. This is provided as metadata to the generator, which merges it with its knowledge stream developed through training and outputs a natural language answer based on the retrieved context.

This decoupling allows the system to handle much longer documents than the LLM's context window would usually permit by focusing attention on only the most relevant passages.

Due to the rapid advancements of LLMs and the constant release of new models, literature applying and evaluating them in depth is limited. However, recent publications demonstrate that RAG-enhanced LLMs can achieve human-level or better performance in complex, domain-specific fields like medicine (Ke et al., 2024) and industrial maintenance (Nagy et al., 2025), while significantly increasing response speed.

However, it is still prone to challenges like substantial computational cost, hallucinations, and retrieval errors across multiple documents. Nevertheless, the technique is constantly evolving, with ongoing research, such as PDFTriage (Saad-Falcon et al., 2023), exploring more effective chunking strategies to process longer, more complex documents.

In summary, a LLM enhanced by a RAG pipeline represents the current best approach to DBQA because it offers a scalable and effective framework for handling long, unstructured documents by decoupling information retrieval and answer generation.

Product Prototyping

Based on the findings from my literature review, I propose the current best solution for a DBQA system as a RAG pipeline powered by an LLM.

To implement my prototype efficiently and not reinvent the wheel, I choose to leverage the OpenAI API for its accessibility and efficacy in handling vector embeddings, document chunking, semantic search, and LLM-based generation. This allowed me to focus my time on building the surrounding infrastructure needed to support the workflow, consisting of:

- A web UI to handle file uploads and allow users to submit questions about their content.
- Backend logic using Flask to manage routing, handle document preprocessing (including chunking), and act as an interface between the frontend and the OpenAI platform.
- Vector store management via the OpenAI API, which stores chunks of document embeddings for fast retrieval.
- Query routing and response generation, where user-submitted questions are constructed into API calls that retrieve the most relevant document sections and generate answers via GPT-4o-mini.

The core logic of my solution is adapted from a recent OpenAI guide (Porte, 2025) which outlines a minimal file search and RAG workflow. My prototype extends this implementation by integrating a modern web UI to enhance UX, providing a seamless DBQA solution for users.

I built my product using a lightweight tech stack—Flask for the backend, the OpenAI Python SDK for readable API interaction, and Tailwind CSS for frontend styling. This streamlined toolkit allowed for rapid prototyping and easy iteration, making the system simple to modify, scale, or extend into a complete product in future development phases.

More implementation details and setup instructions can be found in my GitHub repository: https://github.com/eddysfc/openai-rag-dbqa

Success and Limitation Analysis

To evaluate the performance of my prototype, I tested a variety of PDF documents, including research papers, textbooks, and saved web articles. I prompted the system with both extractive questions (e.g., *“What is the title of the document?”*) and abstractive questions (e.g., *“What are the limitations of the method?”*). Generally, the system was able to retrieve relevant context in extractive tasks and generate coherent, contextually-grounded summaries for abstractive tasks. I determined that most answers were correct with regards to the source material and were clearly phrased.

However, performance noticeably deteriorated under certain conditions. Less-structured documents, especially those with key context enclosed within figures and tables, resulted in reduced retrieval accuracy. Furthermore, the model was prone to hallucination when the correct answer required synthesizing information spread across multiple disparate sections of the document. This is likely due to limitations with the default chunking strategy used by the OpenAI API, which splits text based on token windows without considering document structure. As a result, relevant knowledge may be split across separate chunks, making it harder for the retriever to gather the full context.

Quantitatively, I informally measured:

- Response accuracy: Verification of if responses were correct and grounded in document content.
- Response latency: Typically 3–5 seconds for file upload, 5–8 seconds for LLM text generation.
- Common error types: Objective inaccuracy, hallucinations, vague summaries, redundant phrasing.

Overall, the prototype performs effectively for lightweight, extractive QA tasks and can suffice for summarization in abstractive QA tasks. However, it is not yet suitable for production-level deployment, especially in applications with real-world consequences (i.e., in a business setting). Further improvements are needed in robustness, domain-specific understanding, and error transparency (as detailed below).

Future Improvements

To address the limitations observed, several potential improvements could be made in future iterations:

- *Upgrading to a more powerful model:* The current system employs GPT-4o-mini to stay within free-tier usage limits. To improve performance, a larger model with a longer context window (such as GPT-4.1) could be employed. For more logically-demanding abstractive tasks, reasoning models with chain-of-thought-prompting may also help with developing a nuanced contextual understanding.
- *Custom chunking strategy:* Currently, chunking is performed with OpenAI's default token-based chunking. While efficient for prototyping purposes, due to not having to implement a custom algorithm, it can fragment related content across different chunks. Implementing a more effective chunking technique, such as structural chunking (dividing documents based on headings, sections, or paragraph boundaries using PDF metadata) may improve retrieval performance and contextual understanding, though this specific strategy may not extend to all file formats.
- *User feedback integration and fine tuning:* As demonstrated by the effectiveness of reinforcement learning with human feedback (RLHF) in the training of LLMs, incorporating user feedback (i.e., upvotes, ratings, incorrect flags) could help identify weak categories of responses in the system. These could be used to retrain or fine-tune future retrieval or generation components in the pipeline to further performance.

These enhancements could help the system move beyond its prototyping stage and toward a production-grade tool suitable for application in reducing time spent daily on searching for sparsely-distributed information.

References

Chen, D., Fisch, A., Weston, J., Bordes, A. (2017). Reading Wikipedia to Answer Open-Domain Questions. *arXiv preprint* arXiv:1704.00051.

Fraga, N. (2024). Challenging llms beyond information retrieval: Reasoning degradation with long context windows. *Open Science Framework Preprint* https://doi.org/10.20944/preprints202408.1527.v1.

Ke, Y., Jin, L., Elangovan, K., Abdullah, H.R., Liu, N., Sia, A.T.H., Soh, C.R., Tung, J.Y.M., Ong, J.C.L., Ting, D.S.W. (2024). Development and Testing of Retrieval Augmented Generation in Large Language Models -- A Case Study Report. *arXiv preprint* arXiv:2402.01733.

Khan, A.A., Hasan, M.T., Kemell, K.K., Rasku, J., Abrahamsson, P. (2024). Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report. *arXiv preprint* arXiv:2410.15944.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint* arXiv:2005.11401.

Nagy, A., Spyridis, Y., Argyriou, V. (2025). Cross-Format Retrieval-Augmented Generation in XR with LLMs for Context-Aware Maintenance Assistance. *arXiv preprint* arXiv:2502.15604.

Phogat, K.S., Harsha, C., Dasaratha, S., Ramakrishna, S., Puranam, S.A. (2023). Zero-Shot Question Answering over Financial Documents using Large Language Models. *arXiv preprint* arXiv:2311.14722.

Porte, P.-A. (2025, March 11). *Doing rag on pdfs using file search in the responses API*. OpenAI Cookbook. https://cookbook.openai.com/examples/file_search_responses

Saad-Falcon, J., Barrow, J., Siu, A., Nenkova, A., Yoon, D.S., Rossi, R.A., Dernoncourt, F. (2023). PDFTriage: Question Answering over Long, Structured Documents. *arXiv preprint* arXiv:2309.08872.

Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H. (2016). Bidirectional Attention Flow for Machine Comprehension. *arXiv preprint* arXiv:1611.01603.